REMARKS

The claims remaining in the present application are Claims 1-23. The rejections and comments of the Examiner set forth in the Office Action dated March 4, 2004 have been carefully considered by the Applicants. Applicants respectfully request the Examiner to consider and allow the remaining claims.

35 U.S.C. §102

The present Office Action rejected Claims 1-23 under 35 U.S.C. 102(e) as being anticipated by Lethin et al. (U.S. Publication No. 2002/0,147,969). Applicants have reviewed the above cited reference and respectfully submit that the present invention as recited in Claims 1-23, is neither anticipated nor rendered obvious by the Lethin et al. reference.

Independent Claims 1, 10, 16, and 17

Applicants respectfully point out that independent Claim 1, 10, 16, and 17 each recite embodiments of the present invention that include the detection at intervals whether an interpreter or translator is operating for all instruction sequences. Thereafter, the next instruction sequence is executed using an interpreter or translator depending on the detection of the interpreter or translator over the interval.

In particular, independent Claim 1 recites, in part:

> A method of transferring between types of conversion processes in a computer which converts instructions from a target instruction set to a host instruction set comprising the steps of:
>
> . . .
>
> detecting at intervals whether the interpreter or the translator is operating for all instruction sequences;
>
> increasing a count if the interpreter is operating and decreasing the count if the translator is operating; and
>
> changing from interpreting to translating a sequence of target instructions in response to the count reaching a selected maximum. (Emphasis Added). (Emphasis Added)

That is, independent Claim 1 recites that all instruction sequences are detected over an interval to determine a count. The count then determines whether an interpreter or a translator is used to generate host instructions from target instructions.

Also, independent Claim 10 also recites the detection of all instruction sequences over an interval, as follows:

> A method of optimizing execution by a computer which dynamically converts instructions from a target instruction set to a host instruction set comprising the steps of:
>
> . . .
>
> providing means for determining dynamically which conversion process to use to convert each sequence of instructions, said means depending on detecting at intervals which of the instruction conversion processes is operating for all instruction sequences; and

converting a sequence of instructions using a conversion process determined by said means to convert the sequence of instructions. (Emphasis Added).

That is, independent Claim 10 recites a means for detecting which instruction conversion processes are operating for all instruction sequences in order to determine which conversion process to use to convert a sequence of instructions from a target instruction set to a host instruction set.

Further, independent Claim 16 recites, in part:

A method of transferring between types of conversion processes in a computer which converts instructions from a target instruction set to a host instruction set comprising the steps of:

. . .

comparing interpreter usage to translator usage when executing said code morphing software to produce an interpreter usage to translator usage factor, wherein said interpreter usage to translator usage factor is determined by detecting at intervals whether the interpreter or the translator is operating for all instruction sequences; and

changing from interpreting to translating a sequence of target instructions if the interpreter usage to translator usage factor crosses a threshold. (Emphasis Added).

Specifically, independent Claim 16 recites that a translator usage factor is determined by detecting, for all instruction sequences, whether an interpreter or a translator is used. That is, the interpreter usage to translator usage factor is

determined by detecting over intervals whether the interpreter or translator is operating.

In addition, independent Claim 17 recites, in part:

> A method of transferring between types of conversion processes in a computer which converts instructions from a target instruction set to a host instruction set, said method comprising:
>
> . . .
>
> detecting at intervals whether said interpreter or said translator is operating for all instruction sequences;
>
> in response to said detecting, increasing a count if the interpreter is operating and decreasing said count if said translator is operating;
>
> . . .
>
> changing from interpreting to translating said sequence of instructions upon said sequence of instructions next being encountered by said generating. (Emphasis Added).

That is, independent Claim 17 recites that all instruction sequences are detected over an interval to determine a count. The count then determines whether to change from using an interpreter to using a translator in order to generate host instructions from target instructions.

Independent Claims 1, 10, 16, and 17 determine whether a sequence of instructions should be interpreted or translated through a respective interpreter or translator. In order to determine whether an interpreter or a translator should be applied to the instruction sequences, the detection of the

operation of the interpreter or translator for all instruction sequences over an interval is conducted. The decision to interpret or translate is not tied to the number of times that a particular sequence of instructions is executed, but is tied to the number of times that the interpreter and/or translator is operating in executing all instruction sequences over an interval. Making the determination of whether to translate or interpret based on testing, at intervals, whether an interpreter or a translator is operating for all instruction sequences is non-obvious and is not taught or suggested by the cited references.

Applicants respectfully note that the prior art reference, Lethin et al., does not teach or suggest the detection of the use of an interpreter and a translator over an interval for all instruction sequences to determine whether to interpret or translate an instruction sequence, as claimed in independent Claims 1, 10, 16, and 17 of the present invention.

In contrast to independent Claims 1, 10, 16, and 17 of the present invention, the Lethin et al. reference follows the conventional technique of making a determination as to interpret or compile a particular sequence of instructions based on a count of the number of times an instruction or sequence of instructions is executed through interpretation. Specifically, the Lethin et al. reference discloses the

determination of whether to apply an interpreter or translator to a segment of instructions based on the number of times a corresponding branch instruction is executed using interpretation. If that particular branch instruction is interpreted more than a given number of times, and exceeds a threshold, then that segment of instructions which corresponds to the branch instruction is then compiled, or translated using a translator.

The present invention, on the other hand, as claimed in independent Claims 1, 10, 16, and 17 recites that the determination of whether to interpret or translate the next instruction sequence is determined by detecting the interpretation and translation of all instruction sequences over an interval. That is, embodiments of the present invention detect the number of times an interpreter and translator is used for the execution of all instruction sequences over an interval to determine whether to interpret or translate the next instruction sequence.

In particular, instead of only counting the number of times a particular branch instruction is executed using an interpreter as in the Lethin et al. reference, independent Claim 1 of the present invention recites that all instruction sequences are detected over an interval to determine a count. The count then determines whether an interpreter or a translator is used to generate host instructions from target

instructions. That is, all instruction sequences over an interval are detected to determine whether to interpret or translate when executing the next instruction sequence.

Also, instead of only counting the number of times a particular branch instruction is executed using an interpreter as in the Lethin et al. reference, independent Claim 10 recites that a means is provided for detecting which instruction conversion processes (interpretation or translation) are operating for all instruction sequences in order to determine which conversion process (interpretation or translation) to use to convert a sequence of instructions from a target instruction set to a host instruction set.

Further, instead of only counting the number of times a particular branch instruction is executed using an interpreter as in the Lethin et al. reference, independent Claim 16 recites that a interpreter usage to translator usage factor is determined by detecting for all instruction sequences whether an interpreter or a translator is used. That is, the interpreter usage to translator usage factor is determined by detecting over intervals whether the interpreter or translator is operating when executing all instruction sequences. When the interpreter usage to translator usage factor exceeds a certain threshold, then the next sequence of target instructions is translated instead of interpreted.

Furthermore, instead of only counting the number of times a particular branch instruction is executed using an interpreter as in the Lethin et al. reference, independent Claim 17 recites that all instruction sequences are detected to determine a count of the number of times an interpreter is used versus the number of times a translator is used to execute all instruction sequences over an interval. The count then determines whether to change from using an interpreter to using a translator in order to generate host instructions from target instructions. That is, all instruction sequences over an interval are detected to determine whether to interpret or translate when executing the next instruction sequence.

Thus, Applicants respectfully submit that embodiments of the present invention are not anticipated or suggested by the Lethin et al. reference. Specifically, Applicants respectfully submit that the present invention as disclosed in independent Claim 1 is not anticipated by the Lethin et al. reference, and is in a condition for allowance. In addition, Applicants respectfully submit that Claims 2-9 which depend from independent Claim 1 are also in a condition for allowance as being dependent on an allowable base claim.

Further, Applicants respectfully submit that the present invention as disclosed in independent Claim 10 is not

anticipated or suggested by the Lethin et al. reference, and is in a condition for allowance. In addition, Applicants respectfully submit that Claims 11-15 which depend from independent Claim 10 are also in a condition for allowance as being dependent on an allowable base claim.

Moreover, Applicants respectfully submit that the present invention as disclosed in independent Claim 16 is not anticipated or suggested by the Lethin et al. reference, and is in a condition for allowance.

Additionally, Applicants respectfully submit that the present invention as disclosed in independent Claim 17 is not anticipated or suggested by the Lethin et al. reference, and is in a condition for allowance. Applicants respectfully submit that Claims 18-23 which depend from independent Claim 17 are also in a condition for allowance as being dependent on an allowable base claim.

CONCLUSION

In light of the facts and arguments presented herein, Applicants respectfully request reconsideration of the rejected Claims.

Based on the arguments presented above, Applicants respectfully assert that Claims 1-23 overcome the rejections

of record. Therefore, Applicants respectfully solicit allowance of these Claims.

The Examiner is invited to contact Applicants' undersigned representative if the Examiner believes such action would expedite resolution of the present Application.

Respectfully submitted,

Wagner, Murabito & Hao LLP

Date: 22 June 2004

Lin C. Hsu
Reg. No.: 46,315
Two North Market Street
Third Floor
San Jose, California 95113